

[illegible]

(2)	49	HISTORY
(3)	61	DECLARATIONS
(4)	118	OTSSCVT_TO_L - Convert text (octal) to long
(5)	207	OTSSCVT_TZ_L - Convert text (hex) to long
(6)	284	OTSSCVT_TB_L - Convert text (binary) to long
(7)	357	Local Subroutines

OT Sy
BY CA
ER ER
ER ER
EX EX
EX EX
FO FO
GE IN
OT OT
OT OT
RE SE
TA VA
V_

PS
--
_0

Ph
--
In
Co
Pa
Sy
Pa
Sy
Ps
Cr
As

Th
65
Th
46
0

```
0000 1      .TITLE OTSSCVTTOL      : Convert text to integer
0000 2      .IDENT /1-004/         : File: OTSCVTTOL.MAR Edit: SBL1004
0000 3
0000 4
0000 5 :*****
0000 6 :
0000 7 :*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 :*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 :*  ALL RIGHTS RESERVED.
0000 10 :
0000 11 :*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 :*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 :*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 :*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 :*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 :*  TRANSFERRED.
0000 17 :
0000 18 :*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 :*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 :*  CORPORATION.
0000 21 :
0000 22 :*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 :*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24 :
0000 25 :
0000 26 :*****
0000 27 :
0000 28
0000 29 : FACILITY: Language independent support library
0000 30 :++
0000 31 : ABSTRACT:
0000 32 :
0000 33 : This module contains routines to convert text representations of
0000 34 : integers to longword (or other) values. The text can be in
0000 35 : hexadecimal, octal or binary bases.
0000 36 :
0000 37 :--
0000 38 :
0000 39 : VERSION 1
0000 40 :
0000 41 : HISTORY:
0000 42 :
0000 43 : AUTHOR:
0000 44 : Steven B. Lionel, 21-Feb-1979: Version 1
0000 45 :
0000 46 : MODIFIED BY:
0000 47 :
```


OTSSCVTTOL
1-004

; Convert text to integer
HISTORY

G 11

16-SEP-1984 00:30:37 VAX/VMS Macro V04-00
6-SEP-1984 11:13:53 [LIBRTL.SRC]OTSCVTTOL.MAR;1

Page 2
(2)

```
0000 49      .SBTTL HISTORY
0000 50
0000 51
0000 52 : Edit History for OTSSCVT_TO_L
0000 53 :
0000 54 : 1-001 - Original. Complete rewrite replacing FOR$CNV_IN_0 and
0000 55 :          FOR$CNV_IN_2. SBL 21-Feb-1979
0000 56 : 1-002 - Add OTSSCVT_TB_L. SBL 6-Nov-1980
0000 57 : 1-003 - Fix undetected overflow bug. SBL 4-Jan-1982
0000 58 : 1-004 - Give error if character greater than 127. SPR 11-52485
0000 59 :
```

```
0000 61      .SBTTL  DECLARATIONS
0000 62
0000 63
0000 64      : INCLUDE FILES: None
0000 65      :
0000 66
0000 67
0000 68      : EXTERNAL SYMBOLS:
0000 69      :
0000 70      .DSABL  GBL      ; Prevent undefineds from being
0000 71      ; global.
0000 72      .EXTRN  OTSS_INPCONERR ; Input conversion error
0000 73
0000 74
0000 75      : MACROS: None
0000 76
0000 77
0000 78
0000 79      : PSECT DECLARATIONS:
0000 80      :
0000 81      .PSECT  _OTSSCODE      PIC,SHR,LONG,EXE,NOWRT
0000 82
0000 83
0000 84      : EQUATED SYMBOLS:
0000 85      :
0000 86      REGMASK = ^M<IV,R2,R3,R4,R5,R6> ; Register save mask
0000 87
0000 88
0000 89      : OWN STORAGE:
0000 90      :
0000 91
0000 92      TABLE:
0000 93      : Converts characters to value
00 0000 93      .BYTE  0      : '0'
01 0001 94      .BYTE  1      : '1'
02 0002 95      .BYTE  2      : '2'
03 0003 96      .BYTE  3      : '3'
04 0004 97      .BYTE  4      : '4'
05 0005 98      .BYTE  5      : '5'
06 0006 99      .BYTE  6      : '6'
07 0007 100     .BYTE  7      : '7'
08 0008 101     .BYTE  8      : '8'
09 0009 102     .BYTE  9      : '9'
FF 000A 103     .BYTE -1      : invalid character
FF 000B 104     .BYTE -1      : invalid character
FF 000C 105     .BYTE -1      : invalid character
FF 000D 106     .BYTE -1      : invalid character
FF 000E 107     .BYTE -1      : invalid character
FF 000F 108     .BYTE -1      : invalid character
FF 0010 109     .BYTE -1      : invalid character
0A 0011 110     .BYTE 10      : 'A'
0B 0012 111     .BYTE 11      : 'B'
0C 0013 112     .BYTE 12      : 'C'
0D 0014 113     .BYTE 13      : 'D'
0E 0015 114     .BYTE 14      : 'E'
0F 0016 115     .BYTE 15      : 'F'
0017 116
```



```
0017 118 .SBTTL OTSS$CVT_TO_L - Convert text (octal) to long
0017 119
0017 120 :++
0017 121 : FUNCTIONAL DESCRIPTION:
0017 122 :
0017 123 : Converts a text representation of a base 8 value to internal
0017 124 : form. The usual result of this routine is one longword, but
0017 125 : values of any length (that is a multiple of 8 bits) may be
0017 126 : produced.
0017 127 :
0017 128 : The valid characters for base 8 conversion are -space- and
0017 129 : '0' through '7'. No sign is permitted.
0017 130 :
0017 131 : For compatibility with previous releases, the name
0017 132 : FOR$CNV_IN_0 is equivalent to OTSS$CVT_TO_L.
0017 133 :
0017 134 :
0017 135 : CALLING SEQUENCE:
0017 136 :
0017 137 : status.wlc.v = OTSS$CVT_TO_L (in_string.rt.ds, value.wx.r
0017 138 : [, byte_count.rl.v
0017 139 : [, caller_flags.rl.v])
0017 140 :
0017 141 : INPUT PARAMETERS:
0017 142 :
00000004 0017 143 : in_string = 4 ; Input string by descriptor
0000000C 0017 144 : byte_count = 12 ; Optional count of bytes in
0017 145 : ; value, defaults to 4 if
0017 146 : ; omitted or zero.
00000010 0017 147 : caller_flags = 16 ; Optional.
00000000 0017 148 : V_BN = 0 ; If set, blanks are ignored;
0017 149 : ; otherwise blanks are zeroes.
0017 150 :
0017 151 : IMPLICIT INPUTS: None
0017 152 :
0017 153 : OUTPUT PARAMETERS:
0017 154 :
00000008 0017 155 : value = 8 ; Output value by reference.
0017 156 : ; Length is dependent on
0017 157 : ; byte_count.
0017 158 :
0017 159 : IMPLICIT OUTPUTS: None
0017 160 :
0017 161 : COMPLETION CODES:
0017 162 :
0017 163 : SS$ NORMAL - Successful completion
0017 164 : OTSS$INPCONERR - Input conversion error. Either an illegal
0017 165 : ; character or overflow happened.
0017 166 :
0017 167 : SIDE EFFECTS: None
0017 168 :
0017 169 : --
0017 170 :
0017 171 :
0017 172 : FOR$CNV_IN_0:: ; For compatibility
407C 0017 173 : .ENTRY OTSS$CVT_TO_L, REGMASK
0019 174
```

```

; Convert text to integer
OTSSCVT_TO_L - Convert text (octal) to L

009B 30 0019 175 BSBW SETUP ; Get parameters and set up
                                ; registers.
00D1 30 001C 176 10$: BSBW GET_CHAR ; Get a character
38 19 001F 178 BLSS EXIT_0 ; Exit if done
2E 13 0021 179 BEQL 20$ ; Skip if zero
08 50 91 0023 180 CMPB R0, #8 ; Invalid character?
34 18 0026 181 BGEQ ERROR_0 ; If so, error
03 56 D1 0028 182 CMPL R6, #3 ; Is there room for this digit?
1F 18 002B 183 BGEQ 15$ ; Yes
01 56 D1 002D 184 CMPL R6, #1 ; 0, 1 or 2 bits left?
2A 19 0030 185 BLSS ERROR_0 ; 0 or less is error
0C 14 0032 186 BGTR 12$ ; 2 bits
01 50 91 0034 187 CMPB R0, #1 ; 1 bit - digit 1?
23 14 0037 188 BGTR ERROR_0 ; No, error
63 01 54 50 F0 0039 189 INSV R0, R4, #1, (R3) ; Insert bit
11 11 003E 190 BRB 20$ ; Continue
03 50 91 0040 191 12$: CMPB R0, #3 ; 2 bits - less than 4?
17 14 0043 192 BGTR ERROR_0 ; No, error
63 02 54 50 F0 0045 193 INSV R0, R4, #2, (R3) ; Insert bits
05 11 004A 194 BRB 20$ ; Continue
63 03 54 50 F0 004C 195 15$: INSV R0, R4, #3, (R3) ; Insert bits
54 03 C0 0051 196 20$: ADDL #3, R4 ; Increment bit position
56 03 C2 0054 197 SUBL #3, R6 ; Decrement bit counter
C3 11 0057 198 BRB 10$ ; Loop back for more digits
0059 199
0059 200 EXIT_0: BRW EXIT
004C 31 0059 201
005C 202
005C 203 ERROR_0: BRW ERROR
004D 31 005C 204
005F 205
```



```
005F 207 .SBTTL OTSS$CVT_TZ_L - Convert text (hex) to long
005F 208
005F 209 :++
005F 210 : FUNCTIONAL DESCRIPTION:
005F 211 :
005F 212 : Converts a text representation of a base 16 value to internal
005F 213 : form. The usual result of this routine is one longword, but
005F 214 : values of any length (that is a multiple of 8 bits) may be
005F 215 : produced.
005F 216 :
005F 217 : The valid characters for base 16 conversion are -space-,
005F 218 : '0' through '9' and 'A' through 'F'. No sign is permitted.
005F 219 : Lower case 'a' through 'f' are acceptable.
005F 220 :
005F 221 : For compatibility with previous releases, the name
005F 222 : FOR$CNV_IN_Z is equivalent to OTSS$CVT_TZ_L.
005F 223 :
005F 224 :
005F 225 : CALLING SEQUENCE:
005F 226 :
005F 227 : status.wlc.v = OTSS$CVT_TZ_L (in_string.rt.ds, value.wx.r
005F 228 : [, byte_count.rl.v
005F 229 : [, caller_flags.rl.v])
005F 230 :
005F 231 : INPUT PARAMETERS:
005F 232 :
00000004 005F 233 : in_string = 4 ; Input string by descriptor
0000000C 005F 234 : byte_count = 12 ; Optional count of bytes in
005F 235 : value, defaults to 4 if
005F 236 : omitted or zero.
00000010 005F 237 : caller_flags = 16 ; Optional.
00000000 005F 238 : V_BN = 0 ; If set, blanks are ignored;
005F 239 : otherwise blanks are zeroes.
005F 240 :
005F 241 : IMPLICIT INPUTS: None
005F 242 :
005F 243 : OUTPUT PARAMETERS:
005F 244 :
00000008 005F 245 : value = 8 ; Output value by reference.
005F 246 : Length is dependent on
005F 247 : byte_count.
005F 248 :
005F 249 : IMPLICIT OUTPUTS: None
005F 250 :
005F 251 : COMPLETION CODES:
005F 252 :
005F 253 : SSS$ NORMAL - Successful completion
005F 254 : OTSS$_INPCONERR - Input conversion error. Either an illegal
005F 255 : character or overflow happened.
005F 256 :
005F 257 : SIDE EFFECTS: None
005F 258 :
005F 259 : --
005F 260 :
005F 261 :
005F 262 : FOR$CNV_IN_Z:: ; For compatibility
407C 005F 263 : .ENTRY OTSS$CVT_TZ_L, REGMASK
```

Address	Op Code	Op Code Hex	Op Code Dec	Instruction	Comments
0053	30	0061	264	BSBW	SETUP
		0061	265		; Get parameters and set up registers.
		0064	266		; Get a character
0089	30	0064	267	10\$: BSBW	GET CHAR
13	19	0067	268	BLSS	EXIT_Z
09	13	0069	269	BEQL	20\$:
56	D5	006B	270	TSTL	R6
10	15	006D	271	BLEQ	ERROR_Z
54	50	006F	272	INSV	R0, R4, #4, (R3)
54	04	0074	273	ADDL	#4, R4
56	04	0077	274	SUBL	#4, R6
	E8	007A	275	BRB	10\$:
		007C	276		
		007C	277	EXIT_Z:	
0029	31	007C	278	BRW	EXIT
		007F	279		
		007F	280	ERROR_Z:	
002A	31	007F	281	BRW	ERROR
		0082	282		

```
0082 284 .SBTTL OTSSCVT_TB_L - Convert text (binary) to long
0082 285
0082 286 :++
0082 287 : FUNCTIONAL DESCRIPTION:
0082 288 :
0082 289 : Converts a text representation of a base 2 value to internal
0082 290 : form. The usual result of this routine is one longword, but
0082 291 : values of any length (that is a multiple of 8 bits) may be
0082 292 : produced.
0082 293 :
0082 294 : The valid characters for base 2 conversion are -space-,
0082 295 : "0" and "1". No sign is permitted.
0082 296 :
0082 297 : CALLING SEQUENCE:
0082 298 :
0082 299 : status.wlc.v = OTSSCVT_TB_L (in_string.rt.ds, value.wx.r
0082 300 : [, byte_count.rl.v
0082 301 : [, caller_flags.rl.v])
0082 302 :
0082 303 : INPUT PARAMETERS:
0082 304 :
0082 305 : in_string = 4 ; Input string by descriptor
0082 306 : byte_count = 12 ; Optional count of bytes in
0082 307 : ; value, defaults to 4 if
0082 308 : ; omitted or zero.
0082 309 : caller_flags = 16 ; Optional.
0082 310 : V_BN = 0 ; If set, blanks are ignored;
0082 311 : ; otherwise blanks are zeroes.
0082 312 :
0082 313 : IMPLICIT INPUTS: None
0082 314 :
0082 315 : OUTPUT PARAMETERS:
0082 316 :
0082 317 : value = 8 ; Output value by reference.
0082 318 : ; Length is dependent on
0082 319 : ; byte_count.
0082 320 :
0082 321 : IMPLICIT OUTPUTS: None
0082 322 :
0082 323 : COMPLETION CODES:
0082 324 :
0082 325 : SS$ NORMAL - Successful completion
0082 326 : OTSS_INPCONERR - Input conversion error. Either an illegal
0082 327 : ; character or overflow happened.
0082 328 :
0082 329 : SIDE EFFECTS: None
0082 330 :
0082 331 : --
0082 332 :
0082 333 :
0082 334 : .ENTRY OTSSCVT_TB_L, REGMASK
0084 335 :
0030 30 0084 336 BSBW SETUP ; Get parameters and set up
0087 337 ; registers.
0066 30 0087 338 10$: BSBW GET_CHAR ; Get a character
16 19 008A 339 BLSS EXIT_B ; Exit if done
0E 13 008C 340 BEQL 20$ ; Skip if zero
```



```

; Convert text to integer
OTSSCVT_TB_L - Convert text (binary) to
01 50 D1 008E 341      CMPL R0, #1      ; Illegal character?
    12 14 0091 342      BGTR ERROR_B     ; Yes
    56 D5 0093 343      TSTL R6          ; Is there room for this digit?
    0E 15 0095 344      BLEQ ERROR_B     ; No.
63 01 54 50 F0 0097 345  INSV R0, R4, #1, (R3) ; Insert bit
    54 D6 009C 346 20$: INCL R4          ; Increment bit position
    56 D7 009E 347      DECL R6          ; Decrement bit counter
    E5 11 00A0 348      BRB 10$          ; Loop back for more digits
        00A2 349
        00A2 350 EXIT_B:
0003 31 00A2 351      BRW EXIT
        00A5 352
        00A5 353 ERROR_B:
0004 31 00A5 354      BRW ERROR
        00A8 355
  
```

```

00A8 357      .SBTTL Local Subroutines
00A8 358
00A8 359 ;+
00A8 360 ; EXIT - Exit successfully
00A8 361 ; -
00A8 362
00A8 363 EXIT:
50 01 D0 00A8 364      MOVL    #1, R0      ; Success status code
04 00AB 365      RET                    ; Return
00AC 366
00AC 367 ;+
00AC 368 ; ERROR - Exit with error
00AC 369 ; -
00AC 370
00AC 371 ERROR:
50 00000000'8F 30 00AC 372      BSBW    SETUP      ; To re-zero the value
D0 00AF 373      MOVL    #OTSS_INPCONERR, R0 ; Error code
04 00B6 374      RET                    ; Return with failure
00B7 375
00B7 376

```



```
00B7 378 ;+
00B7 379 ; SETUP - Perform common initialization
00B7 380 ;
00B7 381 ; 1. Determine if optional parameters are present. Set defaults
00B7 382 ; if any.
00B7 383 ; 2. Zero the value.
00B7 384 ; 3. Set up registers as follows:
00B7 385 ; R1 - Count of characters in string.
00B7 386 ; R2 - Points to one byte beyond string.
00B7 387 ; R3 - Address of value.
00B7 388 ; R4 - Set to zero, indicates current value bit offset.
00B7 389 ; R5 - Flag register, set to value of caller_flags.
00B7 390 ; R6 - Count of bits in value.
00B7 391 ; -
00B7 392 ;
00B7 393 SETUP:
56 04 D0 00B7 394 MOVL #4, R6 ; Initial size is 1 longword
03 6C 91 00BA 395 CMPB (AP), #<byte_count/4> ; Is byte count present?
09 19 00BD 396 BLSS 10$ ; No
0C AC D5 00BF 397 TSTL byte_count(AP) ; Is it zero?
04 15 00C2 398 BLEQ 10$ ; Yes (or neg), use default
56 0C AC D0 00C4 399 MOVL byte_count(AP), R6 ; Get byte count
50 08 AC D0 00C8 400 10$: MOVL value(AP), R0 ; Get value address
60 56 00 6E 00 2C 00CC 401 MOVC5 #0, (SP), #0, R6, (R0) ; Fill with zero
00D2 402 ; R5 gets zero
56 56 03 78 00D2 403 ASHL #3, R6, R6 ; R6 gets bit count
51 04 BC 7D 00D6 404 MOVQ @in_string(AP), R1 ; R1 gets character count
00DA 405 ; R2 gets pointer
51 51 3C 00DA 406 MOVZWL R1, R1 ; Clear all but count
52 51 C0 00DD 407 ADDL R1, R2 ; Position pointer to 1 beyond
00E0 408 ; last character.
53 08 AC D0 00E0 409 MOVL value(AP), R3 ; Value address
54 D4 00E4 410 CLRL R4 ; Bit offset
04 6C 91 00E6 411 CMPB (AP), #<caller_flags/4> ; Are caller_flags present?
04 19 00E9 412 BLSS 20$ ; No
55 10 AC D0 00EB 413 MOVL caller_flags(AP), R5 ; Set up caller flags
05 00EF 414 20$: RSB ; Return
```



```
00F0 416 :++
00F0 417 : GET_CHAR - Get a character
00F0 418 :
00F0 419 : Looks at the next character in the input string, right to
00F0 420 : left. If V_BN is set and character is a blank, it is ignored
00F0 421 : and another character is read. If V_BN is not set and the
00F0 422 : character is a blank, it is converted to zero.
00F0 423 :
00F0 424 : The character is translated to an actual value from 0 to 15.
00F0 425 : If the character is not 0-9 or A-F, it branches to ERROR.
00F0 426 :
00F0 427 : The value obtained is moved to R0. If no characters remain,
00F0 428 : R0 is set to -1. Upon exit, the condition codes are set such
00F0 429 : that branches depending on the value in R0 can be made.
00F0 430 :--
00F0 431 :
00F0 432 GET_CHAR:
51 D5 00F0 433 TSTL R1 : Any characters left?
33 15 00F2 434 BLEQ EOS : No, exit with -1
50 72 9A 00F4 435 MOVZBL -(R2), R0 : Put character in R0
51 D7 00F7 436 DECL R1 : Decrement count
20 50 91 00F9 437 CMPB R0, #^A/ / : Is it a blank?
07 12 00FC 438 BNEQ 10$ : No if not equal
EE 55 00 E0 00FE 439 BBS #V_BN, R5, GET_CHAR : Ignore if V_BN set
50 D4 0102 440 CLRL R0 : Blanks are zero
05 05 0104 441 RSB : Exit with zero
50 30 C2 0105 442 10$: SUBL #^A/0/, R0 : Subtract '0'
21 19 0108 443 BLSS ERROR_G : Error if neg
16 50 D1 010A 444 CMPL R0, #Z^A/F/-^A/0/> : Greater than 'F'?
0F 15 010D 445 BLEQ 20$ : Ok if not greater
50 20 C2 010F 446 SUBL #<^A/a/-^A/A/>, R0 : Try lower case
17 19 0112 447 BLSS ERROR_G : Error if between 'F' and 'P'
11 50 D1 0114 448 CMPL R0, #Z^A/A/-^A/0/> : In between '0' and 'a'?
12 19 0117 449 BLSS ERROR_G : Yes, error
16 50 D1 0119 450 CMPL R0, #Z^A/F/-^A/0/> : Greater than 'f'?
0D 14 011C 451 BGTR ERROR_G : Yes, error
50 FEDD CF40 90 011E 452 20$: MOVB TABLE[R0], R0 : Get actual value
05 19 0124 453 BLSS ERROR_G : Error if negative
05 05 0126 454 RSB : Exit with condition codes set
0127 455
05 01 0127 456 EOS: MNEGL #1, R0 : End of string
05 05 012A 457 RSB : Exit
012B 458
012B 460 ERROR_G: BRW ERROR : Illegal character
FF7E 31 012B 461
012E 462
012E 463
012E 464 .END
```

OTSSCVTTOL
Symbol table

; Convert text to integer

E 12

16-SEP-1984 00:30:37
6-SEP-1984 11:13:53

VAX/VMS Macro V04-00
[LIBRTL.SRC]OTSCVTTOL.MAR;1

Page 13
(9)

BYTE COUNT = 0000000C
CALLER_FLAGS = 00000010
EOS 00000127 R 01
ERROR 000000AC R 01
ERROR_B 000000A5 R 01
ERROR_G 0000012B R 01
ERROR_O 0000005C R 01
ERROR_Z 0000007F R 01
EXIT 000000A8 R 01
EXIT_B 000000A2 R 01
EXIT_O 00000059 R 01
EXIT_Z 0000007C R 01
FOR\$CNV-IN_O 00000017 RG 01
FOR\$CNV-IN_Z 0000005F RG 01
GET_CHAR 000000F0 R 01
IN_STRING = 00000004
OTSSCVT_TB_L 00000082 RG 01
OTSSCVT_TO_L 00000017 RG 01
OTSSCVT_TZ_L 0000005F RG 01
OTSS_INPCONERR ***** X 00
REGMASK = 0000407C
SETUP 000000B7 R 01
TABLE 00000000 R 01
VALUE = 00000008
V_BN = 00000000

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes														
ABS	00000000 (0.)	00 (0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE				
_OTSSCODE	0000012E (302.)	01 (1.)	PIC	USR	CON	REL	LCL	SHR	EXE	RD	NOWRT	NOVEC	LONG				

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.03	00:00:03.11
Command processing	117	00:00:00.32	00:00:03.94
Pass 1	83	00:00:00.63	00:00:03.88
Symbol table sort	0	00:00:00.01	00:00:00.01
Pass 2	88	00:00:00.53	00:00:02.23
Symbol table output	3	00:00:00.02	00:00:00.02
Psect synopsis output	2	00:00:00.01	00:00:00.01
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	324	00:00:01.56	00:00:13.20

The working set limit was 1050 pages.

6504 bytes (13 pages) of virtual memory were used to buffer the intermediate code.

There were 10 pages of symbol table space allocated to hold 25 non-local and 12 local symbols.

464 source lines were read in Pass 1, producing 17 object records in Pass 2.

0 pages of virtual memory were used to define 0 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name

Macros defined

_S255SDUA28:[SYSLIB]STARLET.MLB;2

0

0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=(GLOBAL,TRACEBACK)/LIS=LIS\$:OTSCVTTOL/OBJ=OBJ\$:OTSCVTTOL MSRC\$:OTSCVTTOL/UPDATE=(ENH\$:OTSCVTTOL)

0212 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

